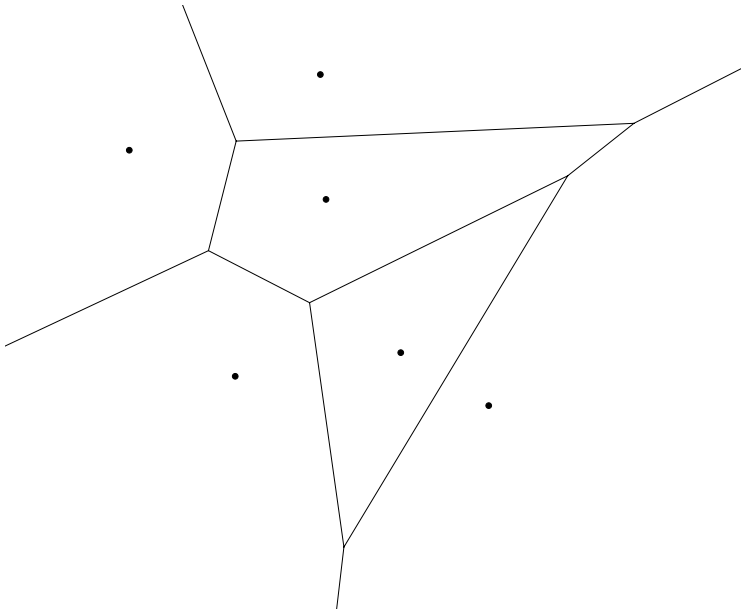
frivol: Määrittelydokumentti

Topi Talvitie

16. toukokuuta 2013

Pistejoukon Voronoi-diagrammilla tarkoitetaan tason jakoa maksimaalisiin alueisiin siten, että kaikilla saman alueen pisteillä on sama lähin piste syötepistejoukosta. Voronoi-diagrammin alueet ovat aina monikulmioita, mahdollisesti rajoittamattomia.



Kuva 1: Esimerkki pistejoukosta ja sen Voronoi-diagrammista.

Projektissa toteutetaan kirjasto annetun pistejoukon Voronoi-diagrammien laskentaan. Kirjaston nimi on *frivol* eli Friendly Voronoi Diagram Library. Tämän lisäksi toteutetaan ohjelma *frivoltool* joka kirjastoa käyttäen laskee annetun pistejoukon Voronoi-diagrammin ja toinen ohjelma *frivoldraw*, joka osaa piirtää kuvan Voronoi-diagrammista.

Syöte ja tuloste

Kirjaston ulkorajapinta on vain yksi funktio, joka ottaa syötteenä taulukon tason pisteitä ja palauttaa taulukon, jossa on vastaavat Voronoi-diagrammin alueet. Alueet palautetaan linkitettynä rakenteena, jossa vierekkäiset alueet, särmät ja solmut on aina linkitetty molempiin suuntiin.

frivoltool on yksinkertainen ohjelma joka käärii kirjaston funktion komentorivikäyttöön sopivaksi, eli syöte saadaan ja tuloste annetaan merkkijonomuodossa.

frivoldraw ottaa syötteen samalla tavalla kuin *frivoltool*, mutta tulosteen kirjoittamisen merkkijonomuodossa sijaan se piirtää siitä kuvan, parametrien mukaan joko suoraan avattavaan ikkunaan tai kuvatiedostoon. Kuvassa näkyy pistejoukko ja Voronoi-diagrammin alueiden reunat.

Algoritmi

frivol laskee pistejoukon Voronoi-diagrammin käyttämällä Fortunen algoritmia, joka vie $O(n \log n)$ aikaa ja $O(n)$ tilaa, kun pisteiden lukumäärä on n. Fortunen algoritmissa pyyhkäisyviiva kulkee tason yli, ja samalla ylläpidetään toista pyyhkäisykäyrää joka koostuu paraabeleista jotka kasvavat muodostamaan diagrammin alueita. Pyyhkäisyviiva kohtaa matkallaan muutostapahtumia, eli uusien solmujen kohtaamisia ja vanhojen paraabelipätkien poistoja. Näitä tapahtumia on $O(n)$ kappaletta. Jokainen tapahtuman käsittely tarvitsee vain $O(1)$ kyselyä/muutosta paraabelikäyrältä, mutta vaatii sen olevan järjestetty.[1][2]

Jos tapahtumien prioriteettijono toteutetaan binäärikeolla, tapahtuman lisäys ja ensimmäisen tapauksen poisto vie $O(\log n)$ aikaa, eli tapahtumajonon käsittely yhteensä $O(n \log n)$ aikaa. Paraabelien ketju voidaan pitää järjestettynä käyttämällä tasapainotettua binääripuuta, kuten AVL-puuta tai punamustapuuta. Koska eri paraabeleja on korkeintaan yksi jokaista syötepistettä kohti, myös paraabelipätkiä on pyyhkäisykäyrällä $O(n)$, eli tapahtumassa käytetään $O(\log n)$ aikaa. Siis myös itse tapahtumien käsittelyyn käytetään yhteensä $O(n \log n)$ aikaa. Molemmissa tietorakenteissa sekä tulosteessa[1] on $O(n)$ alkiota, joten algoritmin muistivaativuuskin on vain $O(n)$.

Prioriteettijono voitaisiin toteuttaa samalla asymptoottisella aika- ja muistivaativuudella myös käyttäen tasapainotettua binääripuuta, jolloin tarvitsisi toteuttaa vähemmän tietorakenteita, mutta binäärikeko on vakiokertoimeltaan huomattavasti nopeampi ja lisäksi hyvin helppo toteuttaa. Binäärikeosta täytyy voida poistaa kesken kaiken solmuja ("false alarms"[2]), mutta tämän voi toteuttaa tekemällä taulukon, joka kuvaa pisteiden indeksit indekseihin taulukkoesityksessä.

Viitteet

[1] S. Fortune, *A sweepline algorithm for Voronoi diagrams*, Proceedings of the second annual symposium on Computational geometry, SCG '86, sivut 313-322. (`http://dl.acm.org/citation.cfm?id=10549`).

[2] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, *Computational geometry*, 2nd revised edition, Springer-Verlag 2000, sivut 151-160.